Effects of Intermittent Faults on the Reliability of a RISC Microprocessor

Joaquín Gracia-Morán, J. Carlos Baraza-Calvo, Daniel Gil-Tomás, Luis J. Saiz-Adalid and Pedro J. Gil-Vicente, *Member, IEEE*

***Abstract*—With the scaling of CMOS technology to the submicron range, designers have to deal with a growing number and variety of fault types. In this way, intermittent faults are gaining importance in modern VLSI circuits. The presence of these faults is increasing due to the complexity of manufacturing processes (which produce residues and parameter variations), together with special aging mechanisms. This work presents a case study of the impact of intermittent faults on the behavior of a RISC microprocessor. We have carried out an exhaustive reliability assessment by using VHDL-based fault injection. In this way, we have been able to modify different intermittent fault parameters, to select various targets, and even, to compare the impact of intermittent faults with those induced by transient and permanent faults.**

***Index Terms*—Fault injection, Hardware description languages, Integrated circuit reliability, Intermittent faults, RISC microprocessor**

I. Introduction

In recent years, the reduction of transistors size has allowed the increase of microprocessors speed and the decrease of their size and supply voltage, but at the cost of augmenting the incidence of faults [1], [2]. This reduction causes a higher rate of transient faults, commonly provoked by temporary environmental conditions (i.e. electromagnetic interferences, cosmic or internal radiation, etc.). Even, radiation may now affect to multiple locations. Also, the changes in the manufacturing processes have increased the rate of permanent faults. This type of faults is produced by irreversible physical changes in a chip. Recently, intermittent faults have emerged as a new source of trouble in deep submicron integrated circuits [3], [4].

Habitually, intermittent faults were considered as a prelude to permanent faults. Wearout processes of an IC usually provoke permanent faults which initially manifest intermittently. Nevertheless, the introduction of new deep submicron technologies makes necessary to study new causes and mechanisms of intermittent faults.

In this way, during last years the effects of such faults in real systems have been analyzed. The failures produced were monitored to determine the most frequent sources of errors and their manifestation [5], [6], [7], [8], [9], [10]. However, the long observation time necessary to perform this type of studies suggests the use of new techniques in order to accelerate the fault occurrence.

Fault injection is a common method to assess the reliability of computer systems [11], [12], [13], [14], [15]. This technique allows a controlled introduction of faults in the system, not being necessary to wait for a long time to log for the apparition of real faults. Fault injection techniques can be classified in three main categories [16]: physical (or Hardware Implemented Fault Injection, HWIFI), software implemented (SWIFI) and simulation-based (SBFI).

Simulation-based Fault Injection has proven to be a good technique to study the impact of intermittent faults [17], [18], [19], [20]. It is a useful experimental way to evaluate the dependability of a system during the design phase. An early diagnosis allows saving costs in the design process, avoiding redesigning in case of error, and thus reducing time-to-market.

Two important issues when using Simulation-based Fault Injection are the accuracy of the system model and the representativeness of the fault models. Regarding this last question, in previous works we have studied some representative causes and mechanisms related to intermittent faults. From this study, we have generated a set of intermittent fault models at logic and register transfer (RT) abstraction levels which can be injected into VHDL models [21].

The objective of this work is to study the impact of intermittent faults in a RISC microprocessor, as well as to compare their consequences with those provoked by permanent and transient faults. To carry out the fault injection experiments, we have used VHDL-based Fault Injection due to its flexibility, as well as the high observability and controllability of the model components [16]. This paper complements previous works published by the authors [18], [21], where the impact of intermittent faults on a commercial CISC microcontroller was analyzed.

The paper is organized as follows. Section II describes the fault injection environment, including a summary of the intermittent fault models applied, a brief description of the intermittent fault parameters, and an outline of the VHDL-

Manuscript received November 9, 2012. This work has been funded by the Spanish Government under the research project TIN2009-13825 and by the Universitat Politècnica de València under the project SP20120806.

All authors are with the Instituto ITACA, Universitat Politècnica de València, Spain.

J. Gracia-Morán mail: Escuela Técnica Superior de Ingeniería Informática (ETSInf), Universitat Politècnica de València, Camino de Vera s\n, 46022 Valencia. E-mail: jgracia@disca.upv.es. Work phone: +34963870000 Ext. 75722. Fax: +34963877579.

E-mail authors: {jgracia, jcbaraza, dgil, ljsaiz, pgil}@disca.upv.es.

based fault injection techniques. Section III depicts the fault injection experiments. Section IV includes a selection of the results. Finally, Section V provides some conclusions.

II. Fault Injection Environment

A. *Intermittent fault models*

Transient and permanent fault models have been traditionally well established, whereas modeling intermittent faults is a pending issue [3]. The most popular fault models for permanent and transient faults are *stuck-at* and *bit-flip*, respectively [16].

Intermittent faults occur due to unstable or marginal hardware. They can be activated by an environmental change such as temperature or voltage alterations. Manufacturing residues, process variations and special wearout processes can also lead to such faults. The introduction of new deep submicron technologies makes necessary to study new fault causes and mechanisms of intermittent faults. Table I summarizes some representative physical causes and fault mechanisms of intermittent faults, as well as the fault models proposed in every case [21]. The table tries to unify, classify and relate the different fault sources. It shows intermittent fault models for buses, storage elements, input/output connections and combinational logic. These fault models are defined at logic and register transfer (RT) abstraction levels. More information can be found in [18], [21].

B. *Intermittent fault parameters*

Intermittent faults manifest as occasional bursts that typically repeat themselves from time to time, and whose effects are not continuous. Also, intermittent faults occur repeatedly in the same places [8]. The duration of intermittent faults is not constant. Instead, it depends on some variable aspects like manufacturing process, environment, wearout process, etc. In this way, the number of times that the fault is active during a burst, as well as the duration of each activation and the separation between activations, have been defined as parameters of the intermittent fault models [22]. Fig. 1 explains the burst parameters.

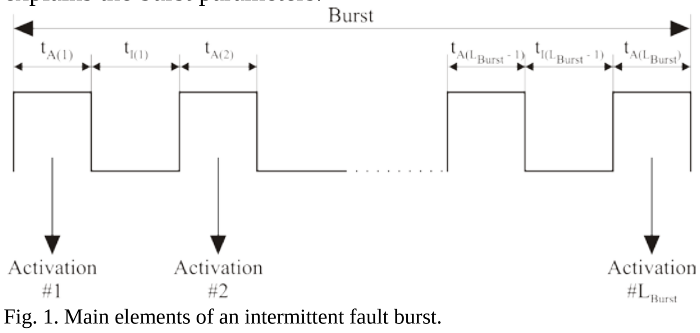


Fig. 1. Main elements of an intermittent fault burst.

C. *Fault injection techniques*

Fig. 2 shows the classification of the different VHDL-based fault injection techniques [16].

With *simulator commands*, it is possible to change, at simulation time, the value or the timing of the signals and variables of the system. *Saboteurs* and *mutants* modify the VHDL code of the system by inserting injection components (*saboteurs*) or activating "mutated" versions of the existing components (*mutants*). Although these two techniques are more complex to apply, and introduce more spatial and temporal overhead than *simulator commands*, they allow injecting more complex fault models. *Other techniques* extend the syntax and semantics of the VHDL language.

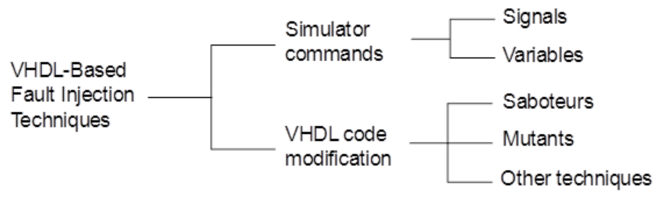


Fig. 2. VHDL-based fault injection techniques.

We have injected the faults by using a tool developed by our research group called VFIT (VHDL-based Fault Injection Tool) [16]. VFIT is able to inject faults automatically applying simulator commands, saboteurs and mutants techniques. The different injection experiments presented in this paper have been carried out with simulator commands, as all fault models selected can be injected by this technique and their application implies lower temporal and spatial overheads.

III. Experiments Descriptions

The main purpose of the study presented in this work is to analyze the influence of intermittent faults in the behavior of a RISC microprocessor. The system target is the Plasma microprocessor [23]. It has a 32-bit MIPS architecture with four-stage pipeline. The VHDL model of Plasma is described at RT and logic abstraction levels.

As workload, Bubblesort sorting algorithm has been used, as it exercises the main elements of the microprocessor: memory, registers, buses, ALU and CU. In this way, we have injected intermittent faults in the storage elements (the register bank and the RAM memory), the buses and the combinational logic of the ALU and CU. Fig. 3 shows the structure of the Plasma core and the injection targets.

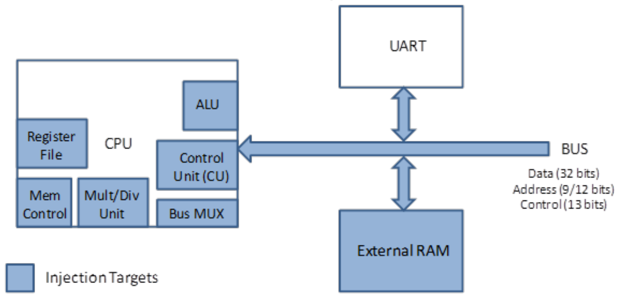


Fig. 3. Block diagram of the Plasma core and injection targets.

TABLE I.
SOME INTERMITTENT FAULT MECHANISMS AND MODELS

Causes	Targets	Fault mechanisms	Type of fault	Fault models
Residues in cells	Memory and registers	Intermittent contacts	Manufacturing defect	*Intermittent stuck-at*
Solder joints	Buses	Intermittent contacts	Manufacturing defect	*Intermittent pulse* *Intermittent short* *Intermittent open*
Electromigration Delamination	Buses I/O connections	Variation of metal resistance Voids	Wearout-Timing	*Intermittent delay* *Intermittent short* *Intermittent open*
Crosstalk	I/O connections Buses	Electromagnetic interference	Internal noise Timing	*Intermittent pulse* *Intermittent delay* *Intermittent speed-up*
Gate oxide soft breakdown	NMOS transistors in SRAM cells	Leakage current fluctuation	Wearout-Timing	*Intermittent delay* *Intermittent indetermination*
Negative bias-temperature instability (NBTI)	PMOS transistors in combinational logic	Increase of transistor threshold voltage V_{TH} Reduction of carrier mobility	Wearout-Timing	*Intermittent delay*
Negative bias-temperature instability (NBTI)	PMOS transistors in SRAM cells	Local mismatches among cell transistors, degradation of static noise margin	Wearout	*Intermittent bit-flip*
Hot-carrier injection (HCI)	NMOS transistors in combinational logic	Increase of transistor threshold voltage V_{TH}	Wearout-Timing	*Intermittent delay*
Low-k dielectric breakdown	Buses I/O connections	Leakage current fluctuation Temperature variations Capacity degradation	Wearout-Timing	*Intermittent delay* *Intermittent short*
Doping profile and gate length deviations	MOS transistors in combinational logic and memory	Deviations in V_{TH} Deviations in operation speed	Manufacturing variations	*Intermittent delay*

The main injection parameters are:

1) *Fault multiplicity*

Due to technology scaling, intermittent faults will likely affect multiple locations [7]. These multiple locations may be adjacent (i.e. neighbor cells in register and memory, neighbor wires in a bus, etc.) or non-adjacent. Thus, we have injected both single and multiple faults.

During the configuration phase of each experiment, we have considered two aspects:

- The number of faults in non-adjacent locations. We have generated the number of non-adjacent targets using a Uniform distribution function in the range [2, N_{NA}], where N_{NA} is the total number of non-adjacent locations.
- In multiple-bit targets, the number of adjacent locations. In this case, we have applied a Uniform distribution function in the range [2, $N_A/2$], where N_A is the target width, that is, the number of adjacent locations.

2) *Fault types*

Intermittent, transient and permanent faults have been injected.

3) *Fault models*

According to Section II.A, the following intermittent fault models have been injected:

- *Intermittent stuck-at*, in storage elements.
- *Intermittent pulse*, in buses.
- *Intermittent* {*pulse*, *open*, *stuck-at*, *indetermination*}, in combinational logic.

The transient fault models injected have been *pulse* (to emulate Single Event Transients, or SETs) in combinational logic and buses; and *bit-flip* (to emulate Single Event Upset, or SEUs) in storage elements. Also, *indetermination* (that is, undefined logic value provoked by voltage and current variations) [16], [24] has been injected in combinational targets.

For permanent faults, the fault models injected have been *stuck-at(0,1)*, *open* and *indetermination* [16], [24].

We have not injected time-related faults (such as the *Intermittent Delay* fault model, see Table I) due to the lack of temporal specifications in the VHDL model. This is usual in core models, as delays are introduced in the implementation phase, after *place and route*.

4) *Burst parameters (for intermittent faults)*

As mentioned in Section II.B, intermittent faults manifest in bursts. So, the following parameters must be configured to inject them (see Fig. 1):

- The burst length (L_{Burst}).
- The activity time (t_A).
- The inactivity time (t_I).

We have generated all the three parameters according to random Uniform distribution functions. For t_A and t_I, three time ranges have been used: [0.01T–0.1T], [0.1T–1.0T], and [1.0T–10.0T], where T is the clock cycle (in our experiments T = 100 ns). L_{Burst} follows a discrete Uniform distribution in the range [1, 10].

5) *Fault duration (for transient faults)*

We have generated the duration of transient faults by using random Uniform distribution functions in three time ranges: [0.01T–0.1T], [0.1T–1.0T], and [1.0T–10.0T].

6) *Injection instant*

We have generated it by using a Uniform distribution function along the workload duration.

7) *Number of faults injected*

In order to obtain a reliable statistic sample, we have injected 1,000 faults per experiment, so that more than 125,000 faults have been injected in total.

8) *Measures obtained*

In order to measure the impact of intermittent faults, we have calculated in every experiment the following percentages:

- Percentage of failures:

$$P_{Failures} = \frac{N_{Failures}}{N_{Injected}} \times 100 \quad (1)$$

- Percentage of latent errors:

$$P_{Latent} = \frac{N_{Latent}}{N_{Injected}} \times 100 \quad (2)$$

- Percentage of non-effective errors:

$$P_{NonEffective} = \frac{N_{NonEffective}}{N_{Injected}} \times 100 \quad (3)$$

where:

- $N_{Injected}$ is the number of faults injected.
- $N_{Failures}$ is the number of failures. A failure is produced when the result obtained after the execution of the workload is erroneous.
- N_{Latent} is the number of latent errors. A latent error is produced when the injected fault propagates to the storage elements but it does not provoke a failure.
- $N_{NonEffective}$ is the number of non-effective errors. An error is non-effective if it does provoke neither failures nor latent errors.

Latent errors and failures are detected by comparing the trace of every fault-injected simulation with a golden run. Fig. 4 summarizes the fault syndrome and the calculated data.

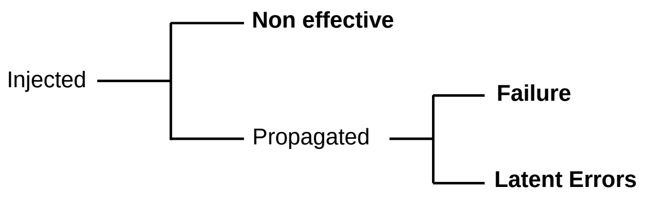


Fig. 4. Fault syndrome.

IV. Results

This section is divided in four parts. Section IV.A analyzes the influence of burst parameters. Section IV.B studies the influence of the injection target. Section IV.C compares the impact of intermittent faults to that of transient and permanent faults. Finally, Section IV.D compares other works where intermittent faults are injected in different microprocessors.

A. Influence of burst parameters

a) Influence of the activity and inactivity times

Fig. 5 represents the impact of intermittent faults in the storage elements. Regarding single faults (see Fig. 5-a), the percentage of failures is very low (~2%), while the percentage of latent errors is high (~53%). This is due to two reasons: i) faults in critical registers mainly provoke failures, independently of the number of activations, and ii) faults in memory mostly cause latent errors because faults are injected randomly in all the memory space, and the workload occupies a very small portion of the memory. On the other hand, as the memory is much bigger than the register bank, the overall behavior tends to that of the memory.

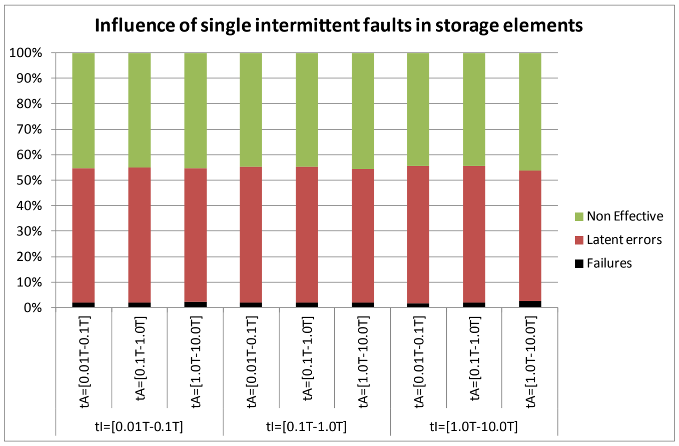
The same trend is observed in multiple faults (see Fig. 5-b). In this case, and as expected, both percentages (failures and latent errors) are higher, with values about 8% and 75% respectively. This is a predictable behavior, as multiple faults affect simultaneously various physical locations of the system.

It is important to emphasize that, in both cases (single and multiple faults), no significant changes are observed when varying t_A and t_I. That is, neither the duration of the activations nor their separation seem to affect $P_{Failures}$ and P_{Latent} in storage elements.

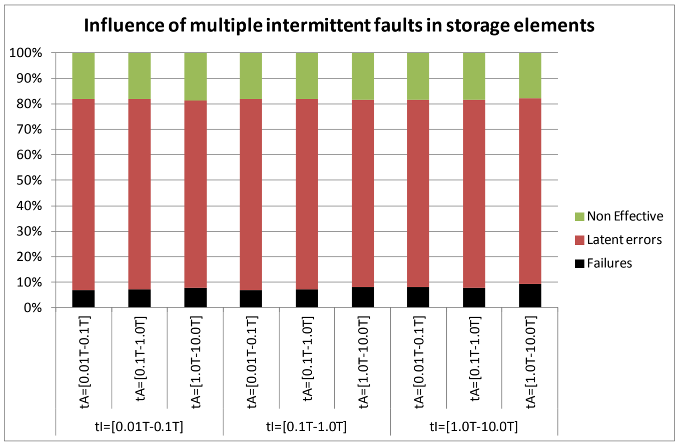
Fig. 6 shows the effects of intermittent faults in buses. As the activity time (t_A) grows:

- The percentage of failures grows appreciably.
- The percentage of latent errors decreases, because faults with longer t_A provoke failures rather than latent errors.
- In general, the system is more affected as the percentage of non-effective errors decreases.

Regarding fault multiplicity, almost all multiple faults affect the system, provoking failures or latent errors (the percentage of non-effective errors is under 2%). A noticeable increment of failures is observed. For the largest t_A, Fig. 6-b shows values of $P_{Failures}$ over 90%.



a)



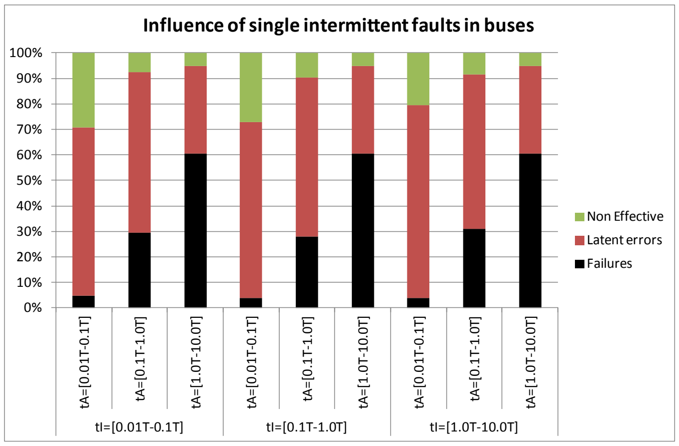
b)

Fig. 5. Influence of intermittent faults in storage elements. a) Single faults. b) Multiple faults.

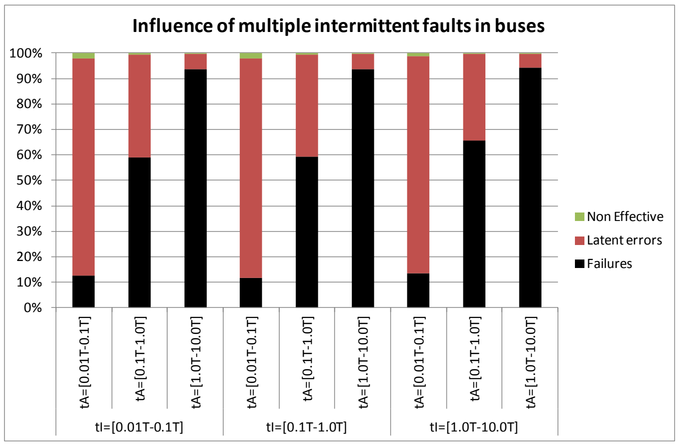
Lastly, Fig. 7 illustrates the effects of intermittent faults in the combinational logic. As in buses, augmenting t_A provokes a clear rise in the percentage of failures.

In single faults, values of $P_{Failures}$ and P_{Latent} are smaller than in buses. This is due to the masking mechanisms existing in combinational logic. As a consequence, $P_{NonEffective}$ is quite bigger. Regarding multiple faults, $P_{Failures}$ is similar to buses, whereas P_{Latent} is smaller, except for the longest values of t_A. Also, like in buses and storage elements, the inactivity time does not have any influence on the results.

In general, we can observe that buses are the most sensitive targets to intermittent faults. Nevertheless, intermittent faults in combinational logic present a non-negligible impact. In multiple faults and for large activity times, their impact can be similar to that in buses.



a)



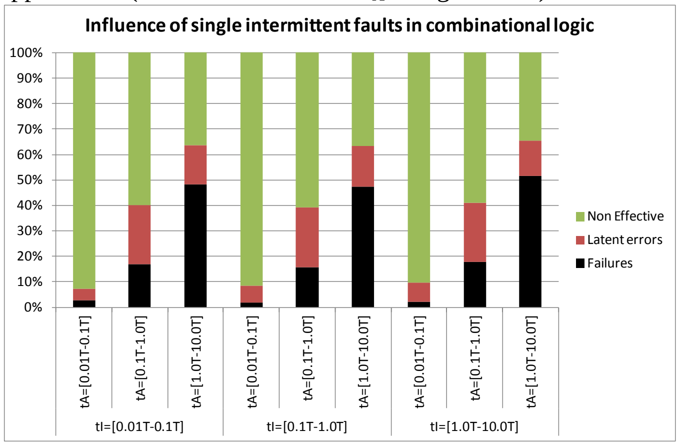
b)

Fig. 6. Influence of intermittent faults in buses. a) Single faults. b) Multiple faults.

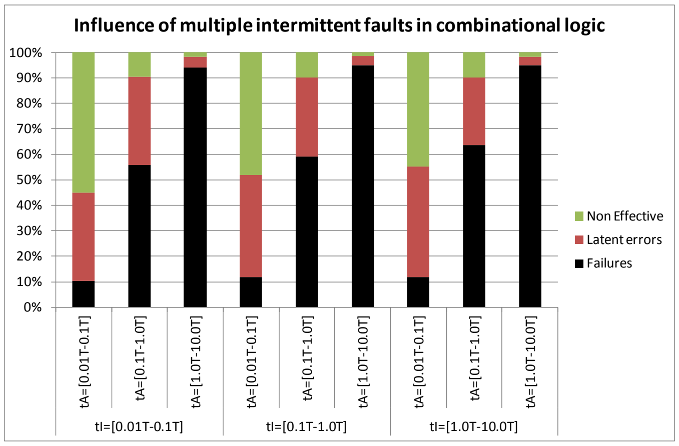
On the other hand, intermittent faults in the storage elements provoke mainly latent errors, with a very low percentage of failures. Unexpectedly, the activity time has no influence on the results. This is due to both the absence of masking effects in the propagation, and the existence of a huge quantity of cells, especially in memory, which while perturbed, they are not accessed later by the workload.

Unexpectedly, t_I does not present a significant influence. A deeper analysis has shown that varying the separation between activations does not change the total number of activations in the bursts, because our particular workload is long enough to fit all activations. Nevertheless, in a general case, this parameter is expected to gain importance because: i) it may affect the system behavior, as it can influence the number of activations, and ii) in a Fault-Tolerant System, the separation between activations can affect the detection and recovery latencies.

As expected, multiple faults impact much more than single faults. This is a predictable behavior, as multiple faults affect simultaneously various physical locations of the system. Percentages of failures over 90% can be seen for intermittent multiple faults in buses and combinational logic. Also, in these two targets, t_A influences notably $P_{Failures}$. Particularly, a roughly logarithmic dependency $(P_{Failure} \approx \log t_A)$ can be appreciated (note that the scale of t_A is logarithmic).



a)



b)

Fig. 7. Influence of intermittent faults in combinational logic. a) Single faults. b) Multiple faults.

b) Influence of the burst length

Fig. 8 shows the results obtained when varying L_{Burst} from 1 to 10, with t_A and t_I defined randomly in the intermediate range [0.1T–1.0T]. The figure shows the results for single and multiple faults and for the three targets: storage elements, buses and combinational logic.

As expected, multiple faults provoke more failures and latent errors than single faults for all targets.

Respect to the storage elements (see Fig. 8-a and Fig. 8-b), $P_{Failures}$ presents a nearly constant behavior, with small variations between 6% and 9%. This is due to:

- Faults affecting critical registers provoke a failure in the very first activations (i.e., for lower values of L_{Burst}), so the total burst length does not matter.

- Faults affecting non-accessed memory cells only cause latent errors, but not failures, even in the presence of multiple activations.

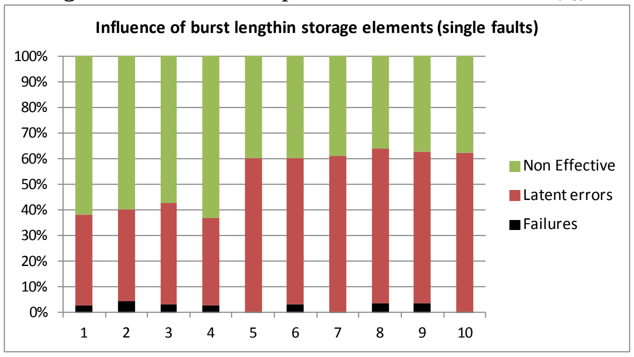
About latent errors, results show that injecting intermittent faults (single or multiple) provokes mainly latent errors. In multiple faults, we can observe a uniform behavior, with variations of P_{Latent} between 70% and 77%. Regarding single faults, we can observe a gap between the values of L_{Burst} 4 and 5, with almost constant values in each interval. Anyway, the values of P_{Latent} are lower. Briefly, L_{Burst} does not influence so much P_{Latent} in storage elements. As faults affect directly the storage cells, errors occurred in the very first activations remain latent.

In buses (see Fig. 8-c and Fig. 8-d), $P_{Failures}$ rises roughly asymptotically. We can approximate $P_{Failures}$ with the exponential dependency $P_{Failures} \approx k(1-e^{-L_{Burst}})$. In single faults, $P_{Failures}$ grows up to 39%, while in multiple faults, the percentage of failures rises up to 75%. In this case, L_{Burst} has a clear influence on $P_{Failures}$. As the number of activations in the same bus wires increases, the probability of fault propagation that provoke a failure augments. From a certain number of activations, the growth is slower and $P_{Failures}$ tends to stabilize.

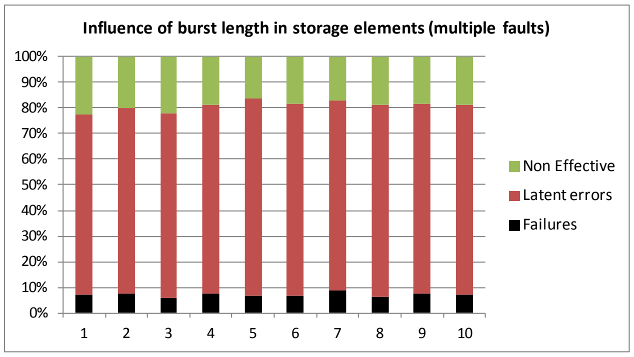
Concerning latent errors in buses, in multiple faults P_{Latent} decreases as L_{Burst} increases, because longer values of L_{Burst} increase $P_{Failures}$. In single faults, P_{Latent} is almost constant.

On the other hand, combinational logic (see Fig. 8-e and Fig. 8-f) presents the same behavior than buses, but with lower values of $P_{Failures}$. In single faults, $P_{Failures}$ grows up to about 28%, and in multiple faults, to 73%. In this type of target, the masking mechanisms are stronger, and thus, as a general trend, the values of $P_{Failures}$ are lower than in buses.

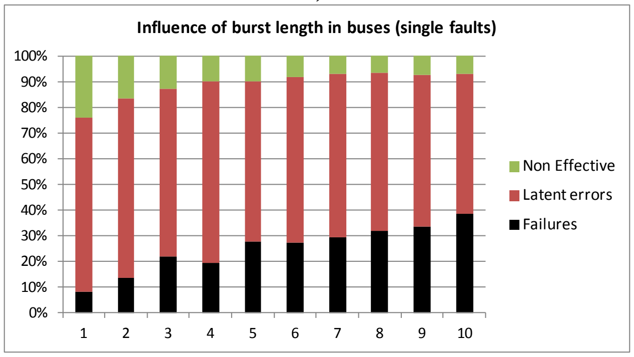
About latent errors, as L_{Burst} increases, their percentage grows slightly in single faults, and decreases in multiple faults. In this last case and as it happened in buses, longer values of L_{Burst} cause higher values of $P_{Failures}$, decreasing in this way P_{Latent}.



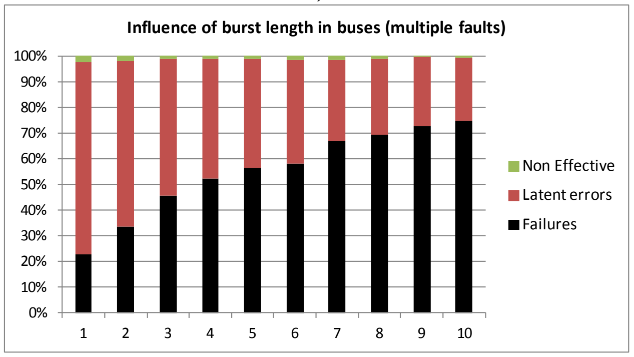
a)



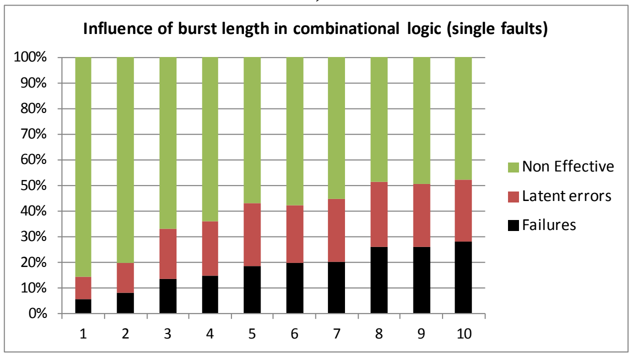
b)



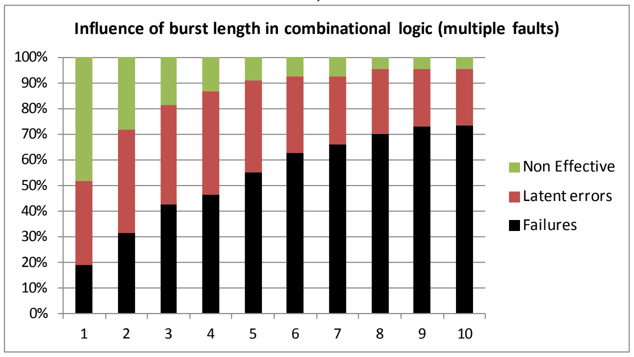
c)



d)



e)



f)

Fig. 8. Influence of the burst length. a) Single faults in storage elements. b) Multiple faults in storage elements. c) Single faults in buses. d) Multiple faults in buses. e) Single faults in combinational logic. f) Multiple faults in combinational logic.

B. Influence of the injection target

From the previous results, it can be inferred that:

- Intermittent faults in buses are very harmful, as buses are used massively in the execution of microprocessor instructions.
- Combinational logic is less sensitive, although the impact of intermittent faults can be notable for high values of t_A and L_{Burst}. The masking mechanisms of this type of logic reduce $P_{Failures}$.
- Intermittent faults in registers provoke a high percentage of failures, because they store intermediate results when executing an instruction. Instead, faults in memory manifest mainly as latent errors.

C. Comparison to transient and permanent faults

In this section, we compare the effects of transient, intermittent and permanent faults in all targets. In these experiments, t_I for intermittent faults has been generated in the intermediate range [0.1T, 1.0T]. As indicated in Section III, t_A for intermittent faults, as well as the duration of transient faults in buses and combinational logic have been generated in three ranges: [0.01T, 0.1T], [0.1T, 1.0T] and [1.0T, 10.0T]. Due to their physical nature, fault duration of transient faults in the storage elements (*bit-flip*) has no sense and thus has not been specified.

Fig. 9 shows the results obtained in storage elements, where an apparently unexpected trend can be noticed. Transient faults provoke more failures and latent errors than intermittent faults. The reason is that there is a low rate of overwrite operations in the memory cells affected by transient faults (*bit-flips*), due to both the memory size and the workload behavior. In this way, these faults present a de facto infinite duration, thus remaining stored permanently. Notice that the intermittent fault model injected in storage elements is the *intermittent stuck-at* (see Table I).

On the other hand, Fig. 10 introduces the results obtained in buses, while Fig. 11 presents the results obtained in combinational logic. In these graphs, the results are the expected. That is, transient faults provoke fewer failures than intermittent faults, as a burst of intermittent faults manifests like a sequence of transient faults in spite of having different origin. As commented before, a consequence of the masking mechanisms of the combinational logic is the lower percentage of latent errors respect to buses. In any case, buses are more sensitive to all fault types, as $P_{NonEffective}$ is lower than in the combinational logic.

In all targets, the greatest impact corresponds to permanent faults because of their infinite duration, although similar values are obtained for the longest values of t_A in intermittent faults.

Fig. 10 and Fig. 11 also show an important dependency on the fault duration of transient faults, similar to that of intermittent faults and the activity time.

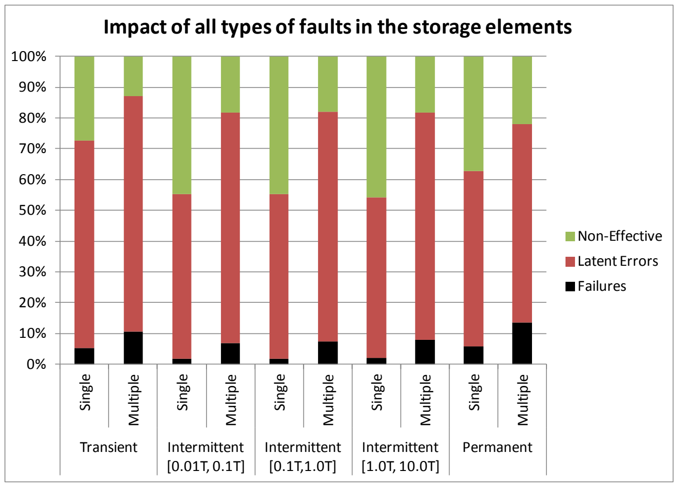


Fig. 9. Comparison of the impact of transient, intermittent and permanent faults in storage elements.

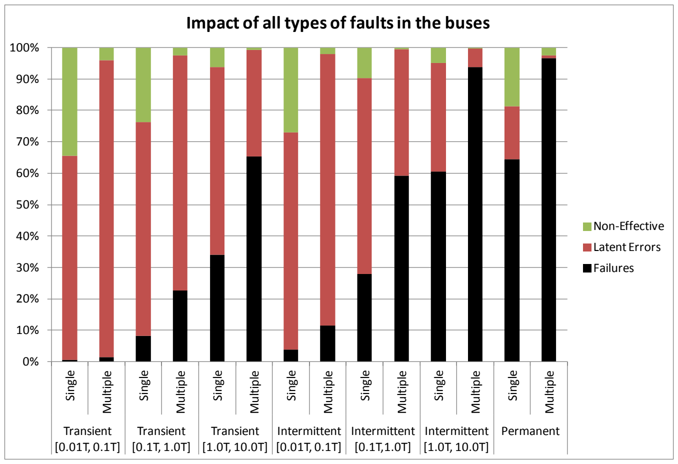


Fig. 10. Comparison of the impact of transient, intermittent and permanent faults in buses.

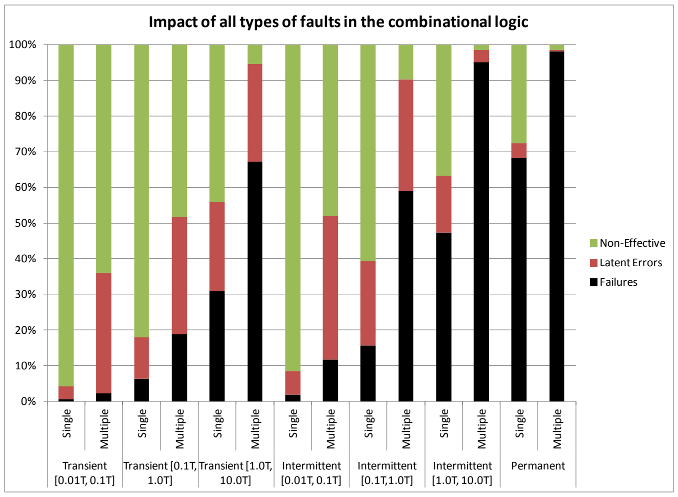


Fig. 11. Comparison of the impact of transient, intermittent and permanent faults in combinational logic.

D. Related work

Present work completes the results presented in [18], [21], where the behavior of an 8051 microcontroller under the influence of intermittent faults is analyzed. Comparing these works with the results presented in this paper, both cores show similar general trends, although some differences have been observed. The Plasma microprocessor is more sensitive to intermittent faults in combinational logic. This is due to the higher complexity of the Plasma in terms of combinational logic (multiplexers, multiplier/divider and the memory controller). Also, more latent errors have been detected in the Plasma, specially caused by faults in the storage modules, mainly because the memory of the Plasma is bigger.

Table II compares the effects of intermittent faults in the Plasma and 8051 cores, summarizing the impact of the different parameters studied in the previous sections. More results about the 8051 core can be seen in [22], [25].

In [26], the impact of transient and intermittent faults on application programs executed in a model of a simple five-stage pipeline RISC processor is compared. The study shows that transient and intermittent faults present substantial differences in the percentage of crashes (failures) caused in programs. Also, it is verified the important influence of the length of the intermittent fault (equivalent to t_A) and its origin (the injection target). That is, the results are similar to those obtained in this paper.

On the other hand, [27] defines a new metric, called IVF (Intermittent Vulnerability Factor), in order to study the impact of intermittent faults in the internal blocks of microprocessors. For the injection experiments, they use a model of the Alpha 21260, a DEC RISC microprocessor. The authors arrive to similar conclusions to those presented in this paper: longer activity times or longer bursts provoke more failures; also, faults in special registers cause a great impact on the system; and finally, intermittent faults provoke more failures in the system than transient faults.

TABLE II.
COMPARISON OF THE EFFECTS OF FAULTS IN THE PLASMA AND 8051 CORES

Parameter influence	**Comparison Plasma vs 8051**
Influence of t_A	Similar trend: • Buses and combinational logic: roughly logarithmic $P_{Failures} \approx \log t_A$ • Storage: negligible
Influence of t_I	Similar trend: negligible (depends on the workload duration)
Influence of L_{Burst}	Similar trend: • Buses and combinational logic: asymptotic rise $P_{Failures} \approx k(1-e^{-L_{burst}})$ • Storage: negligible
Influence of the injection target	• Plasma: buses > combinational logic > storage (except in P_{Latent}) • 8051: buses > storage > combinational logic
Comparison to transient and permanent faults	Similar trend: • Buses and combinational logic: Permanent > intermittent > transient • Storage: Permanent > transient* > intermittent (*) There is a low rate of overwritten cells
Impact of faults in buses	$P_{Failures}$ similar for both cores P_{Latent} similar for both cores
Impact of faults in combinational logic	$P_{Failures}$ higher in Plasma P_{Latent} higher in Plasma
Impact of faults in storage (registers+memory)	$P_{Failures}$ higher in 8051 P_{Latent} higher in Plasma

V. CONCLUSIONS

In this work, we have presented a case study of the effects of intermittent faults on the behavior of a RISC microprocessor. The impact of intermittent faults has been also compared with those provoked by transient and permanent faults. The methodology used lies in VHDL-based fault injection technique, which allows a systematic and exhaustive analysis of the influence of different fault parameters. From the study, some general trends can be extracted:

- The *activity time* is a quite important factor, as increasing the duration of the activations provokes a significant rise of the percentage of failures, especially in buses and combinational logic. A roughly logarithmic growth has been observed. The increase of the *activity time* is a trend in the intermittent faults caused by aging mechanisms. On the other hand, the *inactivity time* has not shown any significant effect because the duration of the workload was long enough to fit all activations.
- The *burst length* has also a notable influence. The percentage of failures grows asymptotically when increasing this parameter. The increase of the burst length is also an expected behavior in the intermittent faults provoked by aging mechanisms.
- Another important factor is the fault spatial multiplicity. Multiple faults provoke a much greater percentage of failures than single faults. This is an important issue because it is expected that, as the feature size of the manufacturing process reduces in deep submicron technologies, the presence of multiple intermittent faults will grow.
- With respect to the injection target, we have found significant differences. Buses are the most sensitive targets to intermittent faults. On the other hand, the impact of faults in combinational logic is also important, even similar to that in buses when injecting multiple intermittent faults with higher activity times and burst lengths. It is foreseen that this impact will grow as the effect of masking mechanisms gets reduced in deep submicron technologies, provoking an increase in their sensitiveness to intermittent faults. Lastly, faults in memory provoke mainly latent errors, while faults in registers cause failures, even in the first activations of the intermittent fault.
- In buses and combinational logic, intermittent faults cause a quite greater percentage of failures than transient faults. Intermittent faults with the long activation times present a similar impact to that of permanent faults, which are the most damaging faults. On the other hand, transient faults in storage elements (*bit-flips*) have shown a greater impact than intermittent faults, because this kind of faults presents a de facto infinite duration if not overwritten.

From the results obtained in this work, it can be inferred the necessity of adding mitigation techniques to deliver fast error detection and correction of intermittent faults, mainly in buses and critical registers. On the other hand, mitigation techniques in combinational logic may be increasingly required.

References

[1] International Technology Roadmap for Semiconductors (ITRS). 2011.

[2] S.L. Jen, J.C. Lu and K. Wang, “A review of reliability research on nanotechnology”, IEEE Transactions on Reliability, Vol. 56, Issue 3, pp. 401-410, 2007.

[3] C. Constantinescu, “Trends and Challenges in VLSI Circuit Reliability”, IEEE Micro, Vol. 23, Issue 4, pp.14-19, 2003.

[4] C. Constantinescu, “Intermittent Faults and Effects on Reliability of Integrated Circuits”, in Procs. Annual Reliability and Maintainability Symposium, Las Vegas, USA, 2008, pp. 370 – 374.

[5] T.Y. Lin and D.P. Siewiorek, “Error Log Analysis: Statistical Modeling and Heuristic Trend Analysis”, IEEE Transactions on Reliability, Vol. 39, Issue 4, pp. 419-432, 1990.

[6] D.P. Siewiorek and R.S. Schwarz, “Reliable computer systems: Design and evaluation” Ed. Digital Press, 3rd Edition, 1998.

[7] C. Constantinescu, “Dependability Benchmarking using Environmental Test Tools”, in Procs. Reliability and Maintainability Symposium, Alexandria, USA, 2005, pp. 567–571.

[8] C. Constantinescu, “Impact of Intermittent Faults on Nanocomputing Devices”, in Procs. DSN 2007 Workshop on Dependable and Secure Nanocomputing, Edinburgh, UK, 2007, pp. 238–241.

[9] J. Guilhemsang, O. Héron, N. Ventroux, O. Goncalves and A. Giulieri, “Impact of the Application Activity on Intermittent Faults in Embedded Systems”, in Procs. 2011 29th IEEE VLSI Test Symposium, Dana Point, California, USA, 2011, pp. 191-196.

[10] E.B. Nightingale, J.R. Douceur and V. Orgovan, “Cycles, Cells and Platters: An Empirical Analysis of Hardware Failures on a Million Consumer PCs”, in Procs. EuroSys 2011, Salzburg, Austria, 2011, pp. 343-356.

[11] J. Arlat, M. Aguera, L. Amat, Y. Crouzet, J.C. Fabre, J.C. Laprie, E. Martins and D. Powell, “Fault Injection for Dependability Validation: A Methodology and Some Applications”, IEEE Transactions on Software Engineering, Vol. 16, Issue 2, pp. 166–182, February 1990.

[12] G.S. Choi, R.K. Iyer, V.A. Carreno, “Simulated Fault injection: A Methodology to Evaluate Fault Tolerant Microprocessor Architectures”, IEEE Transactions on Reliability, Vol. 39, Issue 4, pp. 486-491, 1990.

[13] J. Arlat, M. Aguera, Y. Crouzet, J.C. Fabre, E. Martins, D. Powell, “Experimental Evaluation of the Fault Tolerance of an Atomic Multicast System”, IEEE Transactions on Reliability, Vol. 39, Issue 4, pp. 455-467, 1990.

[14] G. Miremandi and J. Torin, “Evaluating Processor-Behavior and Three Error-Detection Mechanisms Using Physical Fault-Injection”, IEEE Transactions on Reliability, Vol. 44, Issue 3, pp. 441-454, 1995.

[15] D. Avresky, J. Arlat, J.C. Laprie, Y Crouzet, “Fault Injection for Formal Testing of Fault Tolerance”, IEEE Transactions on Reliability, Vol. 45, Issue 3, pp. 443-455, 1996.

[16] A. Benso and P. Prinetto, eds., Fault Injection Techniques and Tools for VLSI reliability evaluation, Kluwer Academic Publishers, 2003.

[17] J. Gracia-Moran, D. Gil-Tomas, L.J. Saiz-Adalid, J.C. Baraza and P.J. Gil-Vicente, “Experimental Validation of a Fault Tolerant Microcomputer System against Intermittent Faults”, in Procs. 40th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, Chicago, USA, 2010, pp. 413-418.

[18] D. Gil-Tomás, J. Gracia-Morán, J.C. Baraza-Calvo, L.J. Saiz-Adalid and P.J. Gil-Vicente, "Analyzing the impact of Intermittent Faults on Microprocessors applying Fault Injection", IEEE Design & Test of Computers, Volume 29, Issue 6, pp. 66–73, 2012.

[19] L. Rashid, K. Pattabiraman and S. Gopalakrishnan, “Modeling the Propagation of Intermittent Hardware Faults in Programs”, in Procs. 16th IEEE Pacific Rim International Symposium on Dependable Computing, Tokyo, Japan, 2010, pp. 19-26.

[20] P.M. Wells, K. Chakraborty and G.S. Sohi, “Adapting to Intermittent Faults in Multicore Systems”, in Procs. 13th International Conference on Architectural Support for Programming Languages and Operating Systems, Seattle, USA, 2008, pp. 255-264.

[21] D. Gil-Tomás, J. Gracia-Morán, J.C. Baraza-Calvo, L.J. Saiz-Adalid and P.J. Gil-Vicente, “Studying the effects of intermittent faults on a microcontroller”, Microelectronics Reliability, Vol. 52, Issue 11, pp. 2387-2846, 2012.

[22] J. Gracia, L.J. Saiz, J.C. Baraza, D. Gil and P.J. Gil, “Analysis of the influence of intermittent faults in a microcontroller”, in Procs. 14th IEEE Design and Diagnostics of Electronic Circuits and Systems, Bratislava, Slovakia, 2008, pp. 80–85.

[23] Plasma CPU model. http://www.opencores.org/projects.mips.

[24] P.J. Gil et al., “Fault Representativeness”, Deliverable ETIE2 of Dependability Benchmarking Project, IST-2000-25245, 2002. Available at: http://www2.laas.fr/TSF/DBench/Deliverables/ETIE2.pdf.

[25] D. Gil, L.J. Saiz, J. Gracia, J.C. Baraza and P.J. Gil, “Injecting Intermittent Faults for the Dependability Validation of Commercial Microcontrollers”, in Procs. IEEE International High Level Design Validation and Test Workshop, Nevada, USA, 2008, pp. 177–184.

[26] J. Wei, L. Rashid, K. Pattabiraman and S. Gopalakrishnan, “Comparing the effects of intermittent and transient hardware faults on programs”, in Procs. 2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops, Hong Kong, China, 2011, pp. 53-58.

[27] S. Pan, Y. Hu and X. Li, “IVF: Characterizing the vulnerability of microprocessor structures to intermittent faults”, in Procs. Design, Automation and Test in Europe, Dresden, Germany, 2010, pp. 238-243.

Joaquín Gracia-Morán is B.Sc. (1995), M.Sc. (1997) and Ph.D. (2004) in Computer Engineering from the Universitat Politècnica de València (UPV). He is currently an associate professor at the UPV, Spain, where he teaches in the Department of Computer Engineering (DISCA). He is member with the Fault-Tolerant Systems (STF) research line within the Institute for the Applications of Advanced Information and Communication Technologies (ITACA). His research interests include design and implementation of digital systems, design and validation of Fault-Tolerant Systems and VHDL-based Fault Injection.

J.Carlos Baraza-Calvo is B.Sc. (1993) and Ph.D. (2003) from the Universitat Politècnica de València, (UPV). Now, he is an associate professor at the UPV, Spain, in the DISCA. He is a member with the STF-ITACA. His research interests include design and implementation of digital systems, design and validation of Fault-Tolerant Systems and Fault Injection.

Daniel Gil-Tomás is an associate professor at the Universitat Politècnica de València, Spain, in the DISCA. He received his B.Sc. degree in Electrical and Electronic Physics from the Universitat de València in 1985. He obtained his Ph.D. degree on Computer Engineering from the UPV in 1999. He is a member with the STF-ITACA. His research interests include design and validation of Fault-Tolerant Systems, Reliability Physics and Reliability of Emerging Nanotechnologies.

Luis J. Saiz-Adalid is M.Sc. (1995) in Computer Engineering from the Universitat Politècnica de València (UPV), and he is currently doing his Ph.D. degree. After 15 years in the industry (IBM, 1995-Celestica, 1998), he is currently a full time lecturer professor at the UPV, Spain. He is also a student member with the STF-ITACA. His research interests include design and implementation of digital systems, design and validation of Fault-Tolerant Systems and VHDL-based Fault Injection.

Pedro J. Gil-Vicente is professor at the Universitat Politècnica de València (UPV), where he has been head of the Department of Computer Engineering (DISCA). Professor Gil has taught courses on Computer Technology, Digital Design, Computer Networks and Fault Tolerant Systems. He is the head of the Fault-Tolerant Systems (STF) research line, within ITACA. His research focuses on the design and validation of real-time fault-tolerant distributed systems, the dependability validation using fault injection, the design and verification of embedded systems, and the dependability and security benchmarking. He has authored more than 90 research papers on these subjects. He has also served as Program Committee member in the IEEE International Conference on Dependable Systems and Networks (DSN), the European Dependable Computing Conference (EDCC) and the Latin American Symposium on Dependable Computing (LADC), and as reviewer in international magazines and congresses related to dependability and security. He was general chair of the EDCC-8 conference, held in Valencia on April 2010.